Quiz Tool

Report Name	Outline Project Specification
Author (User Id)	Michal Goly (mwg2)
Supervisor (User Id)	Chris Loftus (cwl)
Module	CS39440
Degree Scheme	G600 (Software Engineering)
Date	February 6, 2018
Revision	0.1
Status	Draft

1 Project description

Student engagement and live knowledge monitoring are vital in the provision of good quality lecture content in 2018. It is important to make sure audience understands concepts presented, and quizes can help lecturers judge students' understanding in real time.

Aberystwyth University currently uses Qwizdom live polling tool during the provision of some lectures and practical sessions. The university operates under a single license forcing lecturers to book sessions before they can use the tool. Due to human nature, session hijacking occasionally occurs to the bemusement of both students and lecturers. For example, students could be shown biology slides half way through their geography lecture.

This project will focus on the design and development of an in-house built Quiz Tool, enabling multiple lecturers to use it at the same time and potentially making Qwizdom redundant in the future. This ambition can only be achieved if the project is of high quality and its future maintainability is considered at all stages of the design and development.

The Quiz Tool will allow lecturers to login with their university credentials and they will be authenticated using the university LDAP. Lecturers will then be able to upload their PDF lecture slides and create *Lectures* in the system. Each *Lecture* can be then edited and quizes can be added between slides. An example of a quiz would be an "ABC style" question where a student can select one of the options and submit their answer. Once a lecturer is happy with their *Lecture*, he can broadcast it and receive a session key which can be shared with students. Lecture slides will be shown to all students and the lecture can be delivered in a traditional fashion up to the moment a quiz has been embedded between two slides. Students will then be able to answer the question and polling results will be presented in real time to the lecturer. It will be also possible to export the quiz answers as JSON for future analysis.

The tool will be composed of the back-end with an associated database, and two front-ends. One for lecturers and one for students. Finally, the tool will be developed using an agile methodology, adjusted for a single person project.

2 Proposed tasks

The following tasks will be performed on this project:

1. **Choice of the development methodology**: an appropriate methodology has to be chosen for a single person project. A variation of SCRUM could be used, with weekly sprints and sprint planning/retrospectives shorten to the absolute minimum. This would provide the benefits of being able to track the velocity of the project, be able to plan ahead and stay on track easier, without completely "hacking" things together on day to day basis.

2. **Technology considerations**: the MEAN stack will be used to develop the tool. MEAN stack consist of JavaScript based technologies (**M**ongoDB database, **E**xpress minimalistic JavaScript web development framework, **A**ngular 4 front-end framework and **N**odeJS a JavaScript engine). Some prototyping will be necessary to gain familiarity with the tools mentioned above. Furthermore, each part of the app will be containerised using Docker and containers will run together in the same fashion on the developer's machine, the build tool, and in production using docker-compose container orchestration tool.

3. **On-site vs external hosting**: it would be beneficial to keep the web application running within the university intranet. This however, proved impossible with the LXC debian containers available to students for their final year projects. The continuous integration and container deployments mentioned in this document will require more customisation and therefore external cloud provider has to be chosen to allow suitable deployment.

4. **Setting up version control and suitable safeguards**: the code will be stored in a private GitHub repository. There will be a `master` production branch, and a `development` branch. The feature-branch git workflow will be used and each branch will need to have an associated issue (story) on GitHub. Then a pull request will be opened between the `feature-branch` and the `development` and once all tests pass the merge will be allowed. Direct push to both `master` and `development` will be forbidden.

5. **Setting up a continuous integraion build agent**: a suitable CI tool needs to be chosen. Travis, Circle CI and Jenkins are all appropriate for the task at hand. The build agent will checkout the code from the version control, build the Quiz Tool, run all the tests and deploy the tool to production if the branch being built is `master`.

6. **Development**: as previously mentioned, an interative development approach will be used. Development will start with prototyping, followed by the development of "first pass" implementations of all services to get something working as soon as possible. Then the tool can be polished to deliver the final product. It is important to quickly address the most technically difficult parts of the system to avoid problems later on in the process. Special attention needs to be focused on the real time nature of the system and how this should be implemented. Another interesting problem will be the persistance of the MongoDB running in a Docker container, as Docker does not preserve data by design when containers are destroyed.

7. **Project Meetings**: weekly meetings with the supervisor will provide immediate feedback on work done and allow re-adjustment of the approach if necessary to stay on track and deliver the software before the deadline.

3 Project deliverables

The following project deliverables are expected.

1. **Stories and burndown charts**: a backlog of issues grouped into epics and milestones will be visible on GitHub, together with burndown charts and a Kanban board provided by ZenHub. A summary of these will be included in the appendix of the Final Report.

2. **Build definition**: a build script will be produced to configure the continuous integration agent. Depending on the CI tool used, the configuration file will be stored in the version control.

3. **Back-end software**: a Docker containerised web application written in Express, and a MongoDB database persistance layer will be developed. The code will be stored in version control, and submitted for assessment together with the Final Report.

4. **Front-end software**: a Docker containerised Angular 4 front-end will be developed to allow both lecturers and students to use the tool. The code will be stored in version control, and submitted for assessment together with the Final Report.

5. **Development and production environments**: code will be run in a similar fashion both locally on the developer's machine, and in the production environment provided by a cloud provider. This will be possible due to the docker-compose container orchestration tool.

6. **Tests**: a set of tests will be added as software will be developed. These will be stored in version control alongside the rest of the code.

7. **Mid-Project Demonstration PDF presentation**: a PDF presentation will be produced to guide the mid-project demonstration.

8. **Final Report**: this document will be the report and associated appendices. In addition to discussing the work, there will be acknowledgement for any 3rd party libraries, frameworks and tools that are used on the project.